REMARKS

Claims 1-66 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 103(a) Rejection:

The Office Action rejected claims 1-7, 10, 11, 13, 14, 16, 17, 19-21, 24, 25, 27-29, 33-39, 44-46, 48, 49, 51-55, 58, 59, 61-63 and 65 under 35 U.S.C. § 103(a) as being unpatentable over Buckle (U.K. Patent 2,332,288) in view of Jagannathan (U.S. Patent 6,496,871). Claims 8, 9, 41-43, 56 and 57 were rejected as unpatentable over Buckle in view of Jagannathan, and further in view of Graham (U.S. Patent 6,594,700), claims 12, 15, 22, 23, 47 and 64 were rejected as unpatentable over Buckle in view of Jagannathan, and further in view of Edward "Core Jini", pages 405-410 (hereinafter "Edward"), claims 18, 26, 40, 50, 60 and 66 were rejected as unpatentable over Buckle in view of Jagannathan, and further in view of Emmerich "Incremental Code Mobility with XML", page 1-10 (hereinafter "Emmerich"), claims 30-32 were rejected as unpatentable over Buckle in view of Jagannathan, and further in view of the "Official Notice".

In regard to claim 1, contrary to the Examiner's assertion, the cited art fails to teach or suggest converting a current computation state of a process into a data representation language representation of the current computation state and storing the data representation language representation of the current computation state of the process. Instead, Buckle discloses a method in which "it is possible to transmit *binary files or byte code* between agents. The byte code sent in a message may include byte code of an agent itself. Buckle's agent enabling layer utilizes the CORBA Externalization service to record object and object states (i.e. an agent) as a stream of data" (Buckle, page 38, lines 12-14). Thus Buckle teaches that the *binary* byte code of an agent may be included in a message from a location L1 to a location L2.

The examiner refers to Buckle's use of an agent communication language (ACL) and seems to equate an ACL message to a data representation language representation. However, Buckle's ACL is a predefined message format containing specific fields for use when transferring information across a network. Buckle's ACL is specifically not a data representation language. Further, Buckle specifically makes reference to the fact that the "content of an agent communication language message is not restricted in format, so it is possible to transmit *binary* files or byte code" (emphasis added, Buckle, page 38, lines 10-12). Buckle clearly does not teach converting the current computation state of a process into a data representation language representation of the current computation state as asserted by the Examiner. Under Buckle any computation state information remains in a binary format and thus cannot be considered in data representation language representation. In contrast, a data representation language is a specific type of language used for describing data in documents. Data representation languages, such as XML, have been used in the prior art to describe, or represent, data in documents. Clearly, Buckle's binary code in ACL messages is explicitly **not** a **data representation language representation** of current computation state.

Jagannathan also fails to teach converting the current computation state of a process into a data representation language representation of the current computation state. In contrast, Jagannathan teaches that, to migrate an agent, agent control data is sent to the new machine using Java serialization and RMI methods (see Jagannathan, col. 17, lines 24-32, and col. 9, lines 56-67). It is well understood in the art that neither serialization nor RMI techniques use any data representation language representations. In contrast, both serialization and RMI use a binary format.

Additionally, contrary to the Examiner's suggestion, Buckle fails to teach storing the data representation language representation of the current computation state of the process. As shown above, Buckle teaches that the actual byte code of an agent, which is not a data representation language representation, is sent to another location as the content parameter of an ACL message. Buckle also teaches that the "agent code... is stored locally at ... location L2, so that the agent can operate at physical location L2."

(Buckle, col. 39, lines 4-8). The examiner equates this storing of agent code to storing a data representation language representation of the current computation state of a process. However, applicants point out that Buckle specifically refers to storing the agent code and that this storing enables the agent *to operate*. Thus, what Buckle is saving cannot be a data representation language representation. Firstly, it is clearly binary code. Secondly, a data representation language representation of an agent would not be able to operate as required by Buckle without further conversion into the agent's actual byte code.

In response to Applicants' previous presentation of the above argument, the Examiner states "when the code including object and object states is packed into the content parameter of an ACL message string, the code including all object states was converted to ACL message." However, as discussed above, an ACL message is not a data representation language representation. Instead, it follows a predefined message protocol. Additionally, the code (including all object states) was not converted into a data representation language representation when packed into the content parameter of the ACL message. Buckle specifically notes that he is relying upon the free format capabilities of the content parameter of an ACL message to send *binary* data (Buckle, page 38, lines 10-12).

Not only does the cited art (Buckle and Jagannathan) not teach or suggest converting a current computation state of a process into a data representation language representation of the current computation state and storing the data representation language representation of the current computation state of the process, the combination of Buckle and Jagannathan also fails to teach or suggest such functionality. Buckle teaches that the byte code of an agent, including objects and object states, may be included in a message for transmission from a location L1 to a location L2, and then may be stored locally on L2. Jagannathan teaches the migration of agents from one machine to another machine, even during process execution, using serialization. Thus, the combination of Buckle and Jagannathan results in system that includes two different methods of agent migration (that of Buckle and that of Jagannathan) neither of which

would include the converting of a current computation state of a process into a data representation language representations of the current computation state of a process. Additionally, such a combination would not include storing a data representation language representation of the current computation state of a process.

Therefore, the rejection of claim 1 is not supported by the teachings of the cited art and withdrawal thereof is respectfully requested. Similar arguments apply in regard to independent claim 51.

Regarding claim 2, Buckle fails to teach wherein the data representation language representation of the current computation state of the process is stored to a space using a space service, wherein the space is operable to store documents including data representation language documents in the distributed computing environment, and wherein the space service is operable to store and retrieve documents to the space for processes in the distributed computing environment. The Examiner cites Buckle (page 39, lines 1-10), asserting that locally storing received agent code equates to use of a space service. The Examiner is incorrect. First of all, as shown above regarding claim 1, Buckle fails to teach the use of data representation language representations of current computation states of processes. Thus, anything received and stored in Buckle would not be a *data representation language representation* of the current computation state of a process. Additionally, a process *locally* storing received agent code is not the same as using a space service operable to store and retrieve documents to a space for processes in the distributed computing environment. Buckle's locally storing received agent code has absolutely nothing to do with using a space service as recited in Applicants' claim 2.

Thus, the rejection of claim 2 is not supported by the teachings of the cited art and withdrawal thereof is respectfully requested. Similar arguments apply in regard to claims 6, 9, 52, 55, and 57.

Regarding claim 3, Applicants disagree with the Examiner's interpretation of Buckle. Buckle does not teach wherein said storing the data representation language

representation of the current computation state of the process comprises sending the data representation language representation to the space service in one or more messages. In contrast, Buckle teaches sending agent code to the same location that will be executing the received agent code (Buckle, page 38, line 31 – page 39, line 13). The Examiner equates sending the agent code to the code's final destination (where it will be operating) with sending the data representation language representation to a space service. However, Buckle does not describe the receiving location as a space service operable to store and retrieve documents to the space for processes in the distributed computing environment. Nor does Buckle describe any other process retrieving the agent code from the receiving location as would be necessary if Buckle was using a space service. Therefore, the rejection of claim 3 is not supported by the teachings of the cited art and withdrawal thereof is respectfully requested. Similar arguments apply in regard to claims 6, 28, and 53.

Regarding claim 6, the remarks above regarding claim 3 also apply to claim 6. Further, Buckle fails to teach receiving the data representation language representation from the space service in one or more messages. The Examiner argues that receiving agent code comprises receiving the data representation language representation from a space service. However, the Examiner also (in the same office action) contends that storing received agent code is sending the data representation language representation to the space service (Final Office Action regarding claims 2 and 3, see also above remarks regarding claim 3). Buckle clearly teaches, and it is empirically obvious, that agent code is received, **then** locally saved (Buckle, page 39, lines 1-8). So, in order to uphold the Examiner's argument, Buckle would have to receive the agent code from the space service **before** storing it to the space service. Clearly this is incorrect. The rejection of claim 6 is not supported by the teachings of the cited art and withdrawal thereof is respectfully requested.

In regard to claim 19, contrary to the Examiner's assertion, the cited art does not teach converting a current computation state of a process into a data representation language representation of the current computation state and sending the data

representation language representation of the current computation state of the process to a second device. Instead, Buckle discloses a method in which "it is possible to transmit *binary files or byte code* between agents. The byte code sent in a message may include *byte code of an agent* itself. The agent enabling layer utilizes the CORBA Externalization service to record object and object states (ie an agent) as a stream of data." (emphasis added, Buckle, page 38, lines 12-14.) Buckle thus teaches that the byte code of an agent, including objects and object states, may be included in a message. Applicants also assert that Buckle teaches the use of ACL messages, which by definition are not **data representation language representations** of the current computation state (see similar remarks above regarding claim 1).

Jagannathan teaches the migration of agents from one machine to another machine using Java serialization and RMI. (Jagannathan, column 9, lines 56- 67 and column 17, line 50 - column 18, line 23). By definition, Java serialization and RMI do not involve data representation language representations of current computation states of processes. Nowhere does Jagannathan teach suggest *converting* a current computation state of a process into a *data representation language representation* of the current computation state and sending the data representation language representation of the current computation state of the process to a second device.

Not only does the cited art (Buckle and Jagannathan) not teach or suggest converting a current computation state of a process into a data representation language representation of the current computation state and storing the data representation language representation of the current computation state of the process, the combination of Buckle and Jagannathan also fails to teach or suggest converting a current computation state of a process into a data representation language representation of the current computation state and storing the data representation language representation of the current computation state of the process. Buckle teaches that the byte code of an agent, including objects and object states, may be included in a message for transmission from a location L1 to a location L2, and then may be stored locally on L2. Jagannathan teaches the migration of agents from one machine to another machine, even during process

execution, using serialization. Thus, the combination of Buckle and Jagannathan results in system that includes two different methods of agent migration (that of Buckle and that of Jagannathan) neither of which would include the converting of a current computation state of a process into a data representation language representations of the current computation state of a process. Additionally, such a combination would not include storing a data representation language representation of the current computation state of a process.

Therefore, the rejection of claim 19 is not supported by the teachings of the cited art and withdrawal thereof is respectfully requested. Similar arguments apply in regard to independent claims 44 and 61.

In regard to claim 27, contrary to the Examiner's assertion, the cited art does not teach or suggest converting a current computation state of a process into a data representation language representation of the current computation state and sending the data representation language representation of the current computation state of the process to a space service for storage. Instead, Buckle discloses a method in which "it is possible to transmit *binary files or byte code* between agents. The byte code sent in a message may include byte code of an agent itself. The agent enabling layer utilizes the CORBA Externalization service to record object and object states (ie an agent) as a stream of data." (emphasis added, Buckle, page 38, lines 12-14.) Thus, Buckle teaches that the byte code of an agent may be included in a message for transmission from a location L1 to a location L2. Buckle further discloses "the agent is received by a receiving agent at location L2. The agent code, carried by the current parameter is stored locally at the physical resource at location L2, so that the agent *can operate* at physical location L2." (emphasis added, Buckle, page 39, lines 6-8.) Buckle thus teaches that the agent code, and not a data representation language representation of the current computation state, from the message is stored locally on L2 once the message is received on L2. Buckle further teaches the use of content parameters of ACL messages and specifically states that it due to the fact that the content parameter is not restricted in format that allows the transmission of binary files or byte code (Buckle, page 38, lines

10-12). Thus, Buckle is clearly teaching the transfer of binary data (agent code) and not the sending of the data representation language representation of the current computation state of a process.

In addition, Buckle does not teach sending the *data representation language representation* of the current computation state to a space service for storage. Buckle teaches including byte code of an agent in a message, and recording object and object states (i.e. an agent) as a *stream of data*, and storing the agent code on the receiving location of the message so that the agent can operate at the physical location (L2). Buckle does not teach or suggest *converting* the current computation state of the agent, the byte code of the agent, or the objects and object states of the agent to a *data representation language representation* or sending the current computation state of the agent, the byte code of the agent, or the objects and object states of the agent to a space service for storage.

Jagannathan teaches the migration of agents from one machine to another machine, even during process execution using serialization. (Jagannathan, column 17, line 50 - column 18, line 23). Nowhere does Jagannathan teach or suggest *converting* a current computation state of a process into a *data representation language representation* of the current computation state and sending the data representation language representation of the current computation state of the process to a space service for storage.

Not only does the cited art (Buckle and Jagannathan) not teach or suggest converting a current computation state of a process into a data representation language representation of the current computation state and storing the data representation language representation of the current computation state of the process, the combination of Buckle and Jagannathan also fails to teach or suggest converting a current computation state of a process into a data representation language representation of the current computation state and storing the data representation language representation of the current computation state of the process. Buckle teaches that the byte code of an agent,

including objects and object states, may be included in a message for transmission from a location L1 to a location L2, and then may be stored locally on L2. Jagannathan teaches the migration of agents from one machine to another machine, even during process execution, using serialization. Thus, the combination of Buckle and Jagannathan results in system that includes two different methods of agent migration (that of Buckle and that of Jagannathan) neither of which would include the converting of a current computation state of a process into a data representation language representations of the current computation state of a process. Additionally, such a combination would not include sending a data representation language representation of the current computation state of a process to a space service.

Therefore, the rejection of claim 27 is not supported by the teachings of the cited art and withdrawal thereof is respectfully requested.

In regard to claim 41, similar arguments as discussed above for claim 1 apply. Furthermore, contrary to the Examiner's assertion, the cited art does not teach generating an advertisement for the stored data representation language representation.

Graham discloses "[a] service provider protocol adapter servlet [that] listens for service advertising requests...The service provider protocol adapter servlet then converts the service advertisement from a service protocol into a canonical representation of service advertising and stores the advertisement in an internal registry. A client protocol adapter servlet listens for client lookup requests and looks up a corresponding service provider in the internal registry...The client protocol adapter servlet then converts a client request into a canonical representation of the request, which is then used to look up the service required by the client. Once a match has been found, the client protocol adapter servlet brokers the mechanism of client-service provider interaction." (Graham, abstract). Thus, Graham teaches a "service provider protocol adapter servlet" that listens for and intercepts service advertising requests, converts service advertisements from a service protocol into a canonical representation, and stores the canonical representations in an internal registry. Nowhere does Graham teach *generating* an advertisement *for the stored*

data representation language representation. In contrast, Graham teaches converting a service protocol advertisement into a canonical form of advertisement (Graham, Figure 5, item 508, col. 6, lines 38-40). Graham teaches only converting pre-existing advertisements into a different form and thus does not teach the generating an advertisement for the stored data representation language representation.

Not only does the individual cited art (Buckle, Jagannathan and Graham) not teach converting a current computation state of a process into a data representation language representation of the current computation state, storing the data representation language representation of the current computation state of the process, and generating an advertisement for the stored data representation language representation, the combination of Buckle, Jagannathan, and Graham as suggested by the Examiner also fails to teach converting a current computation state of a process into a data representation language representation of the current computation state, storing the data representation language representation of the current computation state of the process, and generating an advertisement for the stored data representation language representation.

Buckle teaches that the byte code of an agent, including objects and object states, may be included in a message for transmission from a location L1 to a location L2, and then may be stored locally on L2. Jagannathan teaches the migration of agents from one machine to another machine, even during process execution, using serialization. Graham teaches a "service provider protocol adapter servlet" that listens for and intercepts service advertising requests, converts service advertisements from a service protocol into a canonical representation, and stores the canonical representations in an internal registry. Thus, the combination of Buckle, Jagannathan, and Graham results in system that includes two different methods of agent migration (that of Buckle and that of Jagannathan) that can also convert pre-existing advertisements into other forms. Such a combination would still not include the converting of a current computation state of a process into a data representation language representations of the current computation state of a process.

Furthermore, There is no motivation within Buckle, Jagannathan, or Graham to modify their systems, either single or in combination, to include converting the current computation state of a process into a data representation language representation of the current computation state, to store the data representation language representation of the current computation state, or to generate an advertisement of the stored data representation language representation.

Therefore, the rejection of claim 41 is not supported by the teachings of the cited art and withdrawal thereof is respectfully requested.

Applicants note that claims 30-32 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Buckle in view of Jagannathan in further view of the "Official Notice". Applicants traverse the Examiner's "Official Notice." It was **not** well known in the prior art for a third device to retrieve a stored data representation language representation of the current computation state of the process from a second device, reconstitute the process at the current computation state from the data representation language representation of the current computation state of the process, and resume execution of the process from the current computation state. Such features were not known in the prior art and none of the references cited by the Examiner suggest these features. The agent migrations in Buckle and Jagannathan do not involve a third device. Moreover, the Examiner's stated motivation of "providing more connections in a large network" makes no sense in the context of Buckle and Jagannathan. Regardless of the number of connections or the size of the network, Buckle and Jagannathan are only concerned with migrating an agent from a first location to a second location. Although Applicants also traversed the "Official Notice" in their previous response, the Examiner still has not provided any evidence to support his "Official Notice." Therefore, this rejection is improper.

Applicants also assert that the rejections of numerous ones of the dependent claims are further unsupported by the cited art. However, since the rejections of each of

the independent claims have been shown to be improper, a further discussion of the rejections of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-47200/RCK.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

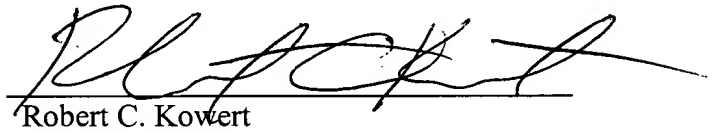
☐ Petition for Extension of Time

☐ Notice of Change of Address

☐ Fee Authorization Form authorizing a deposit account debit in the amount of $ for fees ( ).

☐ Other:

Respectfully submitted,



Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: September 10, 2004